Baumautomaten und Anwendungen

MN

7. März 2005

Inhaltsverzeichnis

Da ich verschiedentlich darauf angesprochen wurde, ob ich diese meine Mitschrift auch veröffentlichen würde, stelle ich sie der Allgemeinheit zur Verfügung.
Wenn ihr (Tipp-) Fehler findet, so schickt sie mir doch bitte mit Kapitel- und Fehlerbeschreibung zu an die folgende eMail-Adresse: *michael.neuendorf@gmx.de*
Ich freue mich über den regen Zuspruch, den ich von Eurer Seite zu diesem Skript bekommen habe. Vielen Dank!

Michael Neuendorf

[2004/10/12]

Ziele der Vorlesung:

1. Erweiterung der Sprachtheorie
2. Lösung logischer Entscheidungsprobleme
3. Model-Checking für verzweigende Systeme
4. Programm-Synthese (endliche zustandsbasierte Systeme)

1 Einleitung

Unendliche Bäume zur Beschreibung von Sachverhalten

Beispiel (Gegenseitiger Ausschluß, Folie 1): Programm als Automat (a, b, c):

- $a \hat{=}$ Zeilennummer Programm 0
- $b \hat{=}$ Zeilennummer Programm 1
- $c \hat{=}$ Variable Turm

Dabei sind die Eigenschaften von Interesse:

(E1) Die Prozesse sind niemals gleichzeitig in der kritischen Region

(E2) Wenn ein Prozeß in die kritische Region möchte, dann erreicht er sie auch irgendwann

- c_0: Prozeß 0 in der kritischen Region $(2, 0, 0), (2, 1, 0), (2, 2, 0), \ldots$
- c_1: Prozeß 1 in der kritischen Region
- r_i: Prozeß i möchte in die kritische Region

Definition (Kripke-Struktur): Sei $P = \{p_1, \ldots, p_n\}$ eine Menge von atomaren Eigenschaften. Eine **Kripke-Struktur** über P hat die Form $\mathcal{K} = (S, R, \lambda)$, wobei

- S ist eine Menge von Zuständen (abzählbar)
- $R \subseteq S \times S$ die Transitionsrelation und
- $\lambda \cdot S \to 2^P$ die Beschriftungsfunktion ist. (Beschriftungsfunktion: $\lambda(s_i) = \{p_i\}$ bedeutet, daß anstelle von s_i p_i gilt.)
- Siehe Folie "Abwicklungsbaum"

Definition (Abwicklung): Sei $\mathcal{K} = (S, R; \lambda)$ uns $s \in S$. Die **Abwicklung** $B(\mathcal{K}, s)$ ist die Kripke-Struktur $B(\mathcal{K}, s) = (S_B, R_B, \lambda_B)$ definiert durch

- $S_B = \{s_0 s_1 \ldots s_x | s_0 = s \wedge (s_i, s_{i+1}) \in R \forall 0 \leq i < k\}$ (Menge aller endlichen Pfade von s_0 aus)
- $R_B = \{(s_0 \ldots s_k, s_0 \ldots s_k s_{k+1}) | (s_k, s_{k+1}) \in R\}$
- $\lambda_B(s_0 \ldots s_k) = \lambda(s_k)$

(E1) und (E2) als Baumeigenschaften

(B1) In dem Abwicklungsbaum kommt kein Zustand mit c_0 und c_1 in der Beschriftung vor.

(B2) Wenn in einem Zustand r_0 gilt, dann gilt auf allen Pfaden, die von diesem Zustand ausgehen, irgendwann c_0. (Analog für Prozeß 1)

Eigenschaften wie (B1) und (B2) werden in Logiken formuliert, deren Modelle Bäume sind. Grundlegende Fragen sind dabei:

Erfüllbarkeit Problem:

- Gegeben: Formel φ
- Frage: Gibt es einen Baum, der Modell von φ ist ?

Model-Checking Problem:

- Gegeben: Formel φ, Kripke-Struktur $\mathcal{K}$ mit Anfangszustand s
- Frage: Ist $B(\mathcal{K}, s)$ Modell von φ ?

Synthese Problem:

- Gegeben: Formel φ
- Aufgabe: Konstruiere eine Kripke-Struktur $\mathcal{K}$ mit Anfangszustand s, sodaß $B(\mathcal{K}, s)$ Model von φ ist.

Die übliche Vorgehensweise ist dabei die folgende: Übersetze φ in einen äquivalenten Automaten $\mathfrak{A}_\varphi$. Die Probleme sind dann:

Leerheit: Akzeptiert $\mathfrak{A}_\varphi$ mindestens einen Baum ?

Zugehörigkeit: Akzeptiert $\mathfrak{A}_\varphi$ den Baum $B(\mathcal{K}, s)$?

Effektive Nichtleerheit: Konstruiere einen Baum, der von $\mathfrak{A}_\varphi$ akzeptiert wird.

1.1 Bäume

Notationen: Für eine Menge X sei

- X^* die Menge der endlichen Folgen/Wörter über X
- X^ω die Menge der unendlichen Folgen/Wörter über X (ω-Wörter)
- $X^+ = X^* \setminus \{\varepsilon\}$, ε ist das leere Wort
- Ein Wort $x \in X^*$ ist Präfix von $y \in X^*$, falls es $z \in X^*$ gibt mit $y = xz$, in Zeichen $x \sqsubseteq y$. Falls $z \neq \varepsilon$, schreibe $x \sqsubset y$.
- Für $\alpha \in X^\omega$, $\alpha = x_0 x_1 x_2 \ldots$ sei $\inf(\alpha) = \{x \in X | x = x_i$ für unendlich viele i
- Ein Alphabet ist eine endliche Menge von Symbolen.
- $[k] = \{1, \ldots, k\}$ für $k \geq 1$

Die Menge $[k]^*$ kann als Baum aufgefaßt werden (im graphentheoretischen Sinne).

Definition: Sei Σ ein Alphabet und $k \geq 1$. Ein Σ-beschrifteter, k-verzweigter Baum ist eine Abbildung $t : [k]^* \to \Sigma$.
Bemerkung:

- Die Σ-beschrifteten, k-verzweigten Bäume entsprechen der Menge Σ^ω.
- Für eine endliche Kripke-Struktur $\mathcal{K} = (S, R, \lambda)$ und $s \in S$ kann man $B(\mathcal{K}, s)$ als $s^P \cap \{\bot\}$-beschrifteten, $|S|$-verzweigten Baum kodieren. ($\bot$: Knoten kommt eigentlich nicht vor)
  O.B.d.A. können wir annehmen, daß $S = [k]$ mit $k = |S|$. Für $x \in [k]^*$ definieren wir

$$t(x) = \begin{cases} \lambda_B(sx) & \text{falls } sx \in S_B \\ \bot & \text{sonst} \end{cases}$$

Ein Pfad π durch einen k-verzweigten Baum ist eine maximale, durch $\sqsubseteq$ geordnete Menge. Die Menge aller dieser Pfade ist Π_k. Für einen Baum $t : [k]^* \to \Sigma$ ist $t(\pi)$ das ω-Wort, das sich durch die Beschriftung entlang des Pfades π ergibt.
Mit $T^\omega_{\Sigma,k}$ bezeichnen wir die Menge aller Σ-beschrifteten, k-verzweigten Bäume. Im Fall der Binärbäume ($k = 2$) schreiben wir T^ω_Σ. Eine Menge $T \subseteq T^\omega_{\Sigma,k}$ heißt **Baumsprache**.

2 Büchi-Baumautomaten

[2004/10/19]

Ein Büchi-Baumautomat (BBA) $\mathfrak{A} = (Q, \Sigma, Q_{in}, \Delta, F)$ über Σ-beschrifteten, k-verzweigten Bäumen besteht aus

- einer endlichen Zustandsmenge Q,
- einer Menge $Q_{in} \subseteq Q$ von Anfangszuständen,
- einer Menge $\Delta \subseteq Q \times \Sigma \times Q^k$ von Transitionen,
- einer Menge $F \subseteq Q$ von akzeptierenden Zuständen.

Für $t \in T^{\omega}_{\Sigma,k}$ ist $\rho \in T^{\omega}_{Q,k}$ ein Lauf von $\mathfrak{A}$ auf t; wenn gilt

- $\rho \in Q_{in}$
- $(\rho(x), t(x), \rho(x1), \ldots, \rho(xk)) \in \Delta$ für alle $x \in [k]^*$.

Ein Lauf ρ ist akzeptierend, wenn auf jedem Pfad durch ρ unendlich viele Endzustände (akzeptierende Zustände) vorkommen.

$$\rho \text{ ist akzeptierend } \Leftrightarrow \forall \pi \in \Pi_k \text{ gilt } \inf(\rho(\pi)) \cap F \neq \emptyset.$$

Ein Baum $t \in T^{\omega}_{\Sigma,k}$ wird von $\mathfrak{A}$ akzeptiert, wenn es einen Lauf von $\mathfrak{A}$ auf t gibt. Die von $\mathfrak{A}$ erkannte (akzeptierte) Baumsprache ist

$$T(\mathfrak{A}) = \{t \in T^{\omega}_{\Sigma,k} | \mathfrak{A} \text{ akzeptiert } t\}$$

$T \subseteq T^{\omega}_{\Sigma,k}$ heißt Büchi-erkennbar, wenn es einen BBA $\mathfrak{A}$ mit $T = T(\mathfrak{A})$ gibt.

Leerheitstest für BBA Das Leerheitsproblem für BBA

- Gegeben: BBA $\mathfrak{A}$
- Frage: Gilt $T(\mathfrak{A}) = \emptyset$?
- Idee zur Lösung: Wir berechnen die Menge aller Zustände, von denen aus ein akzeptierender Lauf existiert.

Für $q \in Q$ ist $\rho \in T^{\omega}_{Q,k}$ ein Lauf von q aus, wenn statt $\rho(\varepsilon) \in Q_{in}$ gilt: $\rho(\varepsilon) = q$. Dazu betrachten wir "endliche Läufe", deren Blätter mit akzeptierenden Zuständen beschriftet sind.
Produktive Zustände: Für $P \subseteq Q$ und $q \in P$ sei $R(q, P)$ die Menge aller Bäume $r \in T_{P,k}$ mit

- $r(\varepsilon) = q$
- Für jedes Blatt x von r ist $r(x) \in P \cap F$.
- Für jeden inneren Knoten x von r gibt es ein $a \in \Sigma$ mit $(r(x), a, r(x1), \ldots, r(xk)) \in \Delta$).

Ein Zustand $q \in P$ ist produktiv bzgl. P, wenn $R(q, P) \neq \emptyset$. Wir nennen P produktiv, wenn alle Zustände $q \in P$ produktiv bzgl. P sind.
Bemerkung:

1. Für $P \subseteq P'$ und $q \in P$ ist $R(q, P) \subseteq R(q, P')$. Insbesondere existiert eine bzgl. Inklusion maximale Menge von produktiven Zuständen $P_{\mathfrak{A}}$.
2. $P_{\mathfrak{A}}$ ist der größte Fixpunkt der Abbildung $f_{pro} : 2^Q \to 2^Q$, $f_{pro}(P) = \{q \in P | q \text{ produktiv bzgl } P\}$.
3. Ist $P_{\mathfrak{A}} \subseteq P$, so ist auch $P_{\mathfrak{A}} \subseteq f_{pro}(P)$. Es gilt $Q \supseteq f_{pro}(Q) \supseteq f_{pro}(f_{pro}(Q)) \ldots = P_{\mathfrak{A}}$

Wir nennen $P_{\mathfrak{A}}$ die Menge der produktiven Zustände von $\mathfrak{A}$. Teil 3 der Bemerkung erlaubt es, $P_{\mathfrak{A}}$ durch Iteration von f_{pro} zu berechnen.

Lemma: Sei $\mathfrak{A} = (Q, \Sigma, Q_{in}, \Delta, F)$ ein BBA. Dann ist $T(\mathfrak{A}) \neq \emptyset$ genau dann, wenn $P_{\mathfrak{A}} \cap Q_{in} \neq \emptyset$ ist.
Zum Beweis brauchen wir das Lemma von König, das für graphentheoretische Bäume formuliert ist:

Lemma von König: Ein endlich verzweigter, unendlicher Baum besitzt einen unendlichen Pfad.
Beweis: Sei B ein endlich verzweigter, unendlicher Baum. Entferne aus B alle Knoten, die nur endlich viele Nachfolger haben. Dann ist B nach dieser Operation nicht leer. Weiterhin hat jetzt Knoten B mindestens einen Nachfolger. B ist immer noch ein Baum, also existiert ein unendlicher Pfad in B.

q.e.d.

Beweis des Lemmas:

- "$\Leftarrow$": Durch Zusammensetzen ausgewählter Element aus den Mengen $R(q, P_{\mathfrak{A}})$ läßt sich ein akzeptierender Lauf von $\mathfrak{A}$ konstruieren.
- "$\Rightarrow$": Sei $\rho \in R^{\omega}_{\mathfrak{A},k}$ ein akzeptierender Lauf von $\mathfrak{A}$. Sei P die Menge der Zustände, die in ρ vorkommen. Wir zeigen, daß P produktiv ist, also $P \subseteq P_{\mathfrak{A}}$. Die Behauptung folgt dann aus $\rho(\varepsilon) \in Q_{in} \cap P$.
  Sei $q \in P$ und $x \in [k]^*$ mit $\rho(x) = q$. Wir definieren einen Graphen B mit Knotenmenge

  $$V_B = \{(y, \rho(y)) \in [k]^* \times Q | y \sqsubseteq x \vee x \sqsubseteq y \vee \forall z \text{ mit } x \sqsubset z \sqsubset y \text{ gilt } \rho(z) \notin F\}$$

  und einer Kante zwischen $(y_1, \rho(y_1))$ und $(y_2, \rho(y_2))$, falls $y_2 = y_1$ für ein $i \in [k]$. Dann hat B folgende Eigenschaften:

  - B ist ein endlich verzweigter Baum.
  - Knoten $(y, \rho(y))$ mit $x \sqsubset y$ und $\rho(y) \in F$ haben keine Nachfolger in B.
  - Jeder unendliche Pfad von B enthält nur endlich viele Zustände aus F.

  Da Pfade durch B Pfaden durch ρ entsprechen und ρ akzeptierend ist, muß B endlich sein. Definiere nun $r \in R(q, P)$ durch

  $$D_r = \{y \in [k]^* | (xy, \rho(xy)) \in V_B\} \text{ und } r(y) = \rho(xy).$$

q.e.d.

Um $P_{\mathfrak{A}}$ zu berechnen, fehlt noch eine Vorschrift, um $f_{pro}(P)$ zu berechnen. Dazu definiere

- $P^{(0)} = P \cap F$
- $P^{(1)} = \{q \in P | \exists q_1, \ldots, q_k \in F \cap P \text{ mit } (q, a, q_1, \ldots, q_k) \in \Delta\}$
- $P^{(i)} = \{q \in P | \exists q_1, \ldots, q_k \in P^0 \cup P^{(i-1)} \text{ mit } (q, a, q_1, \ldots, q_k) \in \Delta\}$

Bemerkung: Für $P \subseteq Q$ ist $q \in P$ produktiv bzgl. P genau dann, wenn $q \in P^{(i)}$ für ein $i \in \mathbb{N}$.

[2004/10/26]

Satz (Algorithmus: Produktive Zustände): Der Algorithmus "Produktive Zustände" berechnet zur Eingabe $\mathfrak{A} = (Q, \Sigma, Q_{in}, \Delta, F)$ die Menge $P_{\mathfrak{A}}$ der produktiven Zustände von $\mathfrak{A}$.

Komplexität: Um P'' zu berechnen, reicht es, jede Transition einmal zu betrachten. P'' kann also in Zeit $O(|\Delta|)$ berechnet werden.
Die innere Schleife wird höchstens $|Q|$ mal durchlaufen. Genauso wird die äußere Schleife nur $|Q|$ mal durchlaufen. Insgesamt ergibt sich eine Laufzeit in $O(|Q|^2 * |\Delta|)$.
Da $L(\mathfrak{A}) \neq \emptyset$ genau dann, wenn $Q_{in} \cap P_{\mathfrak{A}} \neq \emptyset$, erhalten wir:

Satz: Das Leerheitsproblem für BBA ist in Polynomzeit entscheidbar.

3 Propositional Dynamic Logic (PDL)

Idee: Komplexe Programme sind aufgebaut aus atomaren Programmen (z.B. Zuweisungen). Diese atomaren Programme können konkateniert werden oder in Abhängigkeit von Programmzuständen ausgewählt (if) oder iteriert (while) werden. In PDL können solche Konstrukte beschrieben und so Korrektheitsaussagen über Boolesche Programme formuliert werden.

Syntax und Semantik von PDL PDL-Formeln werden über kantenbeschrifteten Kripke-Strukturen interpretiert.

Definition Eine kantenbeschriftete Kripke-Struktur $\mathcal{K} = (S, R, \lambda)$ besteht aus einer Menge S von Zuständen, einer Beschriftungsfunktion $\lambda : S \to 2^P$, wobei P eine Menge von atomaren Propositionen ist, und $R : \Gamma \to 2^{S \times S}$ jedem Symbol aus der Menge Γ von atomaren Programmen eine Transitionsrelation zu. Wir nennen $\mathcal{K}$ deterministisch, wenn es für jedes $b \in \Gamma$ und $s \in S$ maximal ein Paar $(s, s') \in R(b)$ gibt.
PDL-Formeln werden über P und Γ gebildet. Die Menge der Formeln, Test und Programme wird induktiv wie folgt definiert:

- $\top$ ist eine PDL-Formel
- Für jedes $p \in P$ ist p eine Formel
- Für PDL-Formeln φ_1, φ_2 sind $\neg\varphi_1$ und $\varphi_1 \wedge \varphi_2$ PDL-Formeln.
- Ist φ eine PDL-Formel, so ist $\varphi?$ ein Test.
- Für jedes $b \in \Gamma$ ist b ein Programm.
- Ist α ein Programm und φ eine Formel, dann ist $\langle\alpha\rangle\varphi$ eine Formel.
- Ein regulärer Ausdruck α über einem Alphabet Σ aus atomaren Programmen und Test ist ein Programm.
- $\langle(b_1 + b_3)^*\rangle p_1$ *bedeutet, daß das Programm einen Lauf nur aus* b_1 *und* b_3 *hat und im Zustand* p_1 *endet.*
- $\langle b_3; (\neg\langle(b_1 + b_3)^*\rangle p_1)?; b_2\rangle p_2$ *bedeutet, daß das Programm mit* b_3 *beginnt, dann prüft, ob dort der innere Programmteil läuft, und falls dies der Fall ist,* b_2 *laufen läßt und im Zustand* p_2 *endet.*
- while φ do $\alpha \rightsquigarrow (\varphi;\alpha)^*\neg\varphi?$
- if φ then α_1 else $\alpha_2 \rightsquigarrow (\varphi?;\alpha_1 + \neg\varphi?;\alpha_2)$

Mit $\neg$ und $\wedge$ können wir die üblichen Booleschen Operatoren definieren. Weiterhin sei $[\alpha]\varphi \equiv \neg\langle\alpha\rangle\neg\varphi$.

Semantik: Wir definieren induktiv über den Formelaufbau, wann eine PDL-Formel φ in einem Zustand s einer Kripke-Struktur $\mathcal{K}$ gilt, in Zeichen $(\mathcal{K}, s) \models \varphi$, und erweitern parallel dazu die Definition von R auf beliebige Programme und Tests.

- $(\mathcal{K}, s) \models \top$
- $(\mathcal{K}, s) \models p :\Leftrightarrow p \in \lambda(s)$
- $(\mathcal{K}, s) \models \neg\varphi :\Leftrightarrow (\mathcal{K}, s) \not\models \varphi$
- Für eine PDL-Formel φ ist $R(\varphi?) = \{(s, s) | (\mathcal{K}, s) \models \varphi\}$.
- Für ein Programm α über dem Alphabet Σ ist $(s, s') \in R(\alpha)$, wenn es ein Wort $w = a_1 \dots a_n \in L(\alpha)$ und $s_0, \dots, s_n \in S$ gibt mit $s_0 = s, s_n = s'$ und $(s_{i-1}, s_i) \in R(a_i)$ für $i \in \{1, \dots, n\}$ ($a_i \in \Gamma$ oder $a_i = \varphi?$ und $s_{i-1} = s_i$).
- $(\mathcal{K}, s) \models \langle\alpha\rangle\varphi$, falls ein $s' \in S$ existiert mit $(s, s') \in R(\alpha)$ und $(\mathcal{K}, s') \models \varphi$.

Definition (Zustandsmenge einer PDL-Formel): Die durch eine PDL-Formel φ definierte Zustandsmenge in $\mathcal{K}$ ist:

$$\| \varphi \|_{\mathcal{K}} = \{s \in S | (\mathcal{K}, s) \vDash \varphi\}$$

Definition (Ausführbarkeit): Wir sagen, daß ein Programm $b \in \Gamma$ ausführbar ist in einem Zustand s, wenn $(\mathcal{K}, s) \vDash \langle b \rangle \top$ gilt, d.h. es gibt von s aus eine mit b beschriftete Kante.

3.1 Erfüllbarkeit

Wir nennen eine PDL-Formel erfüllbar, wenn es ein Modell zu dieser Formel gibt. Das Erfüllbarkeitsproblem für PDL ist das folgende Entscheidungsproblem:

- Gegeben: PDL-Formel φ
- Frage: Ist φ erfüllbar ?

Beispiel: Bauer, Ziege, Hund, Kohlkopf

- Atomare Propositionen kodieren die Situation:

  $P = \{p_B, p_K, p_Z, p_H\}$
  $\varphi_{start} = \neg p_B \wedge \neg p_K \wedge \neg p_Z \wedge \neg p_H$
  $\varphi_{ziel} = p_B \wedge p_K \wedge p_Z \wedge p_H$

- Atomare Programme für die möglichen Aktionen:

  $\Gamma = \{B, K, Z, H\}$
  $B \hat{=}$ Bauer überquert den Fluß alleine
  $K \hat{=}$ Bauer transportiert Kohlkopf über den Fluß
  $H, Z \hat{=}$ entsprechend

Das Transportproblem ist lösbar gdw. φ erfüllbar:

Deterministisches Erfüllbarkeitsproblem Um Modelle von PDL-Formeln durch k-verzweigte Bäume codieren zu können, werden wir uns zunächst auf determistische Modelle einschränken.
Das deterministische Erfüllbarkeitsproblem ist:

- Gegeben: PDL-Formel φ
- Frage: Hat φ ein deterministisches Modell ?

[2004/11/09]

PDL hat die Baummodelleigenschaft. Ein Baummodell ist ein Paar (B, s), wobei B eine Kripke-Struktur ist, die im graphentheoretischen Sinne ein Baum mit Wurzel s ist.

Satz: Ist φ eine erfüllbare PDL-Formel, so hat φ auch ein Baummodell; es existiert ein Baummodell $(B, s) \vDash \varphi$. Ist φ deterministisch erfüllbar, dann kann B deterministisch gewählt werden.

Beweis: Ist $(\mathcal{K}, s)$ ein Modell von φ, dann ist auch $(B(\mathcal{K}, s), s)$ ein Modell von φ. Formal kann dies per Induktion über den Formelaufbau gezeigt werden.

Notation: Um die Notation zu vereinfachen, setzen wir $\Gamma = \{1, \ldots, k\} = [k]$. Ein deterministisches Baummodell (B, s) läßt sich als k-verzweigter Baum kodieren, indem man jeden Zustand s' mit der Folge der Kantenbeschriftungen des Pfades von s nach s' identifiziert. (s. Folie "Kodierung eines deterministischen Baummodells") Gibt es zu $x \in [k]^*$ keinen Pfad von s aus, der mit x beschriftet ist, dann wird x mit $\perp$ beschriftet. Die Kodierung von (B, s) ist also ein Baum $t : [k]^* \to \Sigma_P := 2^P \cup \{\perp\}$. Für eine PDL-Formel φ schreiben wir $t \vDash \varphi \Leftrightarrow t$ kodiert ein Baummodell (B, s) mit $(B, s) \vDash \varphi$. Wir setzen $T(\varphi) = \{t \in T^{\omega}_{\Sigma_P, k} | t \vDash \varphi\}$.

Ziel: Konstruiere BBA $\mathfrak{A}_\varphi$ mit $T(\mathfrak{A}_\varphi) = T(\varphi)$. Dazu werden wir statt regulärer Ausdrücke NEAs in den Formeln benutzen. Ein Lauf eines BBA ist eine Beschriftung des Eingabebaumes (mit Zuständen), die lokalen Konsistenzbedingungen genügt (den Transitionen). Um $\mathfrak{A}_\varphi$ zu konstruieren, suchen wir eine Möglichkeit, den Eingabebaum t so zu beschriften, daß wir daran erkennen können, ob $t \vDash \varphi$. Wir beschriften jeden Knoten des Baumes mit den Teilformeln, die an diesem Unterbaum gelten.
Dazu sei für $t : [k]^* \to \Sigma_P$ und $x \in [k]^*$ sei t_x der Unterbaum von t an x, definiert durch $t_x(y) = t(xy)$.
Die Menge der benötigten Formeln für die gesuchte Beschriftung ist $\mathrm{cl}(\varphi)$, der Abschluß von φ. (s. Folie "Abschluß $\mathrm{cl}(\varphi)$ einer PDL-Formel")
Wir interessieren uns für die Abbildung $\beta^t_\varphi : [k]^* \to 2^{\mathrm{cl}(\varphi)} \cup \{\bot\}$ mit

$$\beta^t_\varphi = \begin{cases} \{\psi \in \mathrm{cl}(\varphi) | t_x \vDash \psi\} & \text{falls } t(x) \neq \bot \\ \bot & \text{falls } t(x) = \bot \end{cases}$$

Die Eigenschaften von β^t_φ sollen jetzt durch lokale Eigenschaften charakterisiert werden. (s. Folie "φ PDL-Formel ...")
Es gilt, daß β^t_φ ein Beweis von φ in t ist, wenn $t \vDash \varphi$.

Lemma: Sei $t : [k]^* \to \Sigma_P$ die Kodierung eines Baummodells und φ eine PDL-Formel. Ist β ein Beweis von φ in t, dann gilt $\beta = \beta^t_\varphi$. Insbesondere gilt $t \vDash \varphi$ genau dann, wenn es einen Beweis von φ in t gibt.
Die Grundidee ist es, zu φ einen Automaten $\mathfrak{A}_\varphi$ zu konstruieren, dessen Zustände Teilmengen von $\mathrm{cl}(\varphi)$ sind. Ein akzeptierender Lauf von $\mathfrak{A}_\varphi$ auf t soll ein Beweis von φ in t sein. $(B1) - (B4)$ sind bereits lokale Bedingungen, die ein Automat testen kann; $(B5)$ muß noch umformuliert werden.

Satz: Zu einer PDL-Formel φ kann man einen Büchi-Baumautomaten $\mathfrak{A}_\varphi$ mit $T(\varphi) = T(\mathfrak{A}_\varphi)$ konstruieren.

Beweis (siehe Folien): Die Zustände von $\mathfrak{A}_\varphi$ sind die Teilmengen von $\mathrm{cl}(\varphi)$. $\mathfrak{A}_\varphi$ soll so konstruiert werden, daß ein akzeptierender Lauf auf t der Abbildung β^t_φ entspricht.

$$\beta^t_\varphi = \begin{cases} \bot \text{falls } t(x) = \bot \\ \{\psi \in \mathrm{cl}(\varphi) | t_x \vDash \psi\} \end{cases}$$

Zur Charakterisierung von β^t_φ Bedingungen (B1)-(B5). Wir schreiben (B5) in eine lokale Version um, die etwas schwächer ist als (B5). (B5'.b) reicht die zu prüfende Formel an einen Nachfolgeknoten weiter. Um $\langle\alpha\rangle\psi$ zu beweisen, darf das nur endlich oft geschehen. Wir müssen also sicherstellen, daß irgendwann (a) angewendet wird. Dazu wird der BBA α_φ eine zweite Menge von Zuständen verwalten. Diese Menge enthält die Formeln, die durch (B5.b') weitergereicht wurden. Sei $\mathrm{cl}_\diamond(\varphi) = \{\langle\alpha\rangle\psi | \langle\alpha\rangle\psi \in \mathrm{cl}\varphi\}$.
Behauptung: Es gilt $T(\mathfrak{A}_\varphi) = T(\varphi)$. $T(\mathfrak{A}_\varphi) \subseteq T(\varphi)$. Sei ρ ein akzeptierender Lauf von $\mathfrak{A}_\varphi$ auf t und seien ρ_1 und ρ_2 jeweils die Projektionen auf die erste bzw. zweite Komponente der Zustände. Dann erfüllt ρ_1 die Bedingungen (B1)-(B5), ist also ein Beweis von φ in t. Es reicht, (B5) zu beweisen, da (B1)-(B4) wegen (i) aus der Definition von Δ_φ erfüllt. Sei dazu $\langle\alpha\rangle\psi \in \rho_1(x)$. Da ρ akzeptierend ist, wird auf jedem Pfad von x aus irgendwann ein Endzustand erreicht. Sei n_x der maximale Abstand zu einem Endzustand. Wir zeigen jetzt per Induktion über diese n_x, daß (B5) erfüllt ist.

1. Fall: $\langle\alpha\rangle\psi \in \rho_2(x)$
   Dann ist $\rho_2(x) \neq \emptyset$, $\rho(x)$ als nicht in F_φ. Somit ist $n_x \geq 1$. Die Induktion beginnt also bei $n_x = 1$. Dann gilt $\rho_2(xj) = \emptyset$ an allen Nachfolgerzuständen (bzw. $\rho_2(xj) = \bot$). Dann ist das q aus (iii) der Definition von Δ_φ aus F. Somit ist (B5'.a) erfüllt. Somit gilt (B5) an x.
   Ist $n_x > 1$, so kann $q \notin F$ gelten. Dann ist $\langle\alpha_{q'}\rangle\psi \in \rho_2(xj)$. Es gilt $n_{xj} < n_x$. Per Induktion gilt (B5) für $\langle\alpha_{q'}\rangle\psi$ und xj. Mit den anderen Bedingungen aus (iii) erhält man, daß (B5) auch für $\langle\alpha\rangle\psi$ und x gilt.
2. Fall: $\langle\alpha\rangle\psi \notin \rho_2(x)$
   Dann kann man per Induktion mit ähnlicher Argumentation wir im Fall 1 zeigen, daß man nach endlich vielen Schritten "im ersten Fall" landet.
   $T(\varphi) \subseteq T(\mathfrak{A}_\varphi)$: Gilt $t \vDash \varphi$, dann existiert ein Beweis β von φ in t. Konstruiere einen Lauf so, daß die Projektion auf die erste Komponente β entspricht. Dies ist möglich, da (B5') von (B5) impliziert wird.
   Die zweite Komponente des Laufes wird anhand von (B5) definiert.

q.e.d.

Da (B1)-(B5') und die anderen Bedingungen aus der Definition von Δ_φ algorithmisch überprüft werden können, erhalten wir:

Satz: Das deterministische Erfüllbarkeitsproblem für PDL ist in exponentieller Zeit lösbar. (Konstruiere $\mathfrak{A}_\varphi$ und teste auf Leerheit.)

Korollar: Das allgemeine Erfüllbarkeitsprobleme für PDL ist in exponentieller Zeit lösbar. (Es kann in Polynomzeit auf das deterministische Erfüllbarkeitsproblem reduziert werden (siehe Übung).)

4 Muller- und Paritätsbaumautomaten

BBA akzeptieren, wenn auf jedem Pfad unendlich viele Endzustände gesehen werden. Allgemeiner könnte man genau die zulässigen Mengen für $\inf(\rho(\pi))$ angeben. Diese allgemeinen Bedingungen heißen Muller-Bedingungen.

Definition (Muller-Baumautomat): Ein Muller-Baumautomat (MBA) hat die Form $\mathfrak{A} = (Q, \Sigma, Q_{in}, \Delta, \mathcal{F})$ mit Q, Σ, Q_{in} und Δ wie für BBA und $\mathcal{F} \subseteq 2^Q$, d.h. $\mathcal{F} = \{F_1, \ldots, F_n\}$ mit $F_i \subseteq Q$. Ein Lauf ρ von $\mathfrak{A}$ heißt akzeptierend, wenn auf allen Pfaden π gilt: $\inf(\rho(\pi)) \in \mathcal{F}$.
Offensichtlich ist jeder BBA $\mathfrak{A} = (Q, \Sigma, Q_{in}, \Delta, F)$ äquivalent zu dem MBA $\mathfrak{A}' = (Q, \Sigma, Q_{in}, \Delta, \mathcal{F})$ mit $\mathcal{F} = \{S \subseteq Q | S \cap F \neq \emptyset\}$.
Wir werden zeigen, daß $T_b^{<\infty}$ aus Beispiel (a) nicht durch einen BBA erkannt werden kann. Für den Beweis benötigen wir eine Art "Pumping-Lemma" für Büchi-erkennbare Baumsprachen.
Dazu sei zu $t \in T^{\omega}_{\Sigma,k}$ und $x, y \in [k]^*$ der Baum $t^*_{[x,y]}$ definiert durch

$$t^*_{[x,y]}(z) = \begin{cases} t(z) & \text{falls } x \not\sqsubset z \\ t(xz') & \text{falls } z = xy^n z' \text{ mit } n \geq 0 \text{ und } y \not\sqsubset z' \end{cases}$$

[2004/11/22]

Lemma (Pumping-Lemma für BBA): Sei $\mathfrak{A} = (Q, \Sigma, Q_{in}, \Delta, F)$ ein BBA und $t \in T(\mathfrak{A})$. Ist ρ ein akzeptierender Lauf von $\mathfrak{A}$ auf t und sind $x, y, y' \in [k]^*$ mit $\rho(x) = \rho(y)$, $\rho(xy') \in F$, $y \neq \varepsilon$ und $y' \sqsubset y$.

Beweis: $\rho^*_{[x,y]}$ ist ein akzeptierender Lauf auf $t^*_{[x,y]}$.

q.e.d.

Satz: Die Klasse der Büchi-erkennbaren Baumsprachen ist echt in der Klasse der Muller-erkennbaren Baumsprachen enthalten.

Beweis: Angenommen, $T_b^{<\infty}$ (siehe Folie) wird durch den BBA $\mathfrak{A} = (Q, \Sigma, Q_{in}, \Delta, F)$ erkannt. Setze $n = |F|+1$ und definiere t durch

$$t(x) = \begin{cases} b & \text{falls } x \text{ von der Form } (2^*1)^m, m \leq n \\ a & \text{sonst} \end{cases}$$

Dann ist $t \in T_b^{<\infty}$, also $t \in T(\mathfrak{A})$. Sei ρ ein akzeptierender Lauf von $\mathfrak{A}$ auf t. Da ρ akzeptierend ist, existieren $m_1, \ldots, m_n$, sodaß für $x_1 = 2^{m_1}$ und $x_{i+1} = x_i 1 2^{m_{i+1}}$, gilt $\rho(x_i) \in F$ für alle i. Nach der Wahl von n existieren $j < k$ mit $\rho(x_j) = \rho(x_k)$. Da $x_j \sqsubset x_k$ ist, existiert ein $x \neq \varepsilon$ mit $x_k = x_j y$. Dann sind mit $x = x_j$ und $y' = \varepsilon$ die Voraussetzungen des Pumping-Lemmas erfüllt, also ist $t^*_{[x,y]} \in T(\mathfrak{A})$. Es gilt aber $t^*_{[x,y]}(xy^m 1) = b$ für alle $m \in \mathbb{N}$. Da y mit 1 beginnt, ist $xy^m 1 \sqsubseteq xy^{m+1}1$ und somit enthält $t^*_{[x,y]}$ einen Pfad mit unendlich vielen b. Widerspruch zur Annahme, daß $T(\mathfrak{A}) = T_b^{<\infty}$.
Da das Komplement von $T_b^{<\infty}$ Büchi-erkennbar ist, erhalten wir:

Satz: Die Klasse der Büchi-erkennbaren Baumsprachen ist nicht mehr unter Komplement abgeschlossen.
Der Nachweis, daß die Klasse der Muller-erkennbaren Baumsprachen unter Komplement abgeschlossen ist, ist das nächste Ziel der Vorlesung.
Ein großer Nachteil von MBAs ist, daß $\mathcal{F}$ keinerlei Struktur besitzt. Wir führen jetzt eine Normalform ein, bei der $\mathcal{F}$ und $\bar{\mathcal{F}} = s^Q \setminus \mathcal{F}$ unter Vereinigung abgeschlossen sind. Solche Bedingungen lassen sich sehr kompakt als Paritätsbedingungen darstellen.
Ein Paritätsautomat (PBA) hat die Form $\mathfrak{A} = (Q, \Sigma, Q_{in}, \Delta, c)$ mit $c : Q \to \mathbb{N}$. Die Zahl $c(q)$ nennen wir auch Priorität von q. Die Abbildung c werden wir auch auf Mengen bzw. Folgen von Zuständen anwenden. Damit sind die Mengen bzw. Folgen von Prioritäten gemeint, die man durch Anwendung auf die einzelnen Elemente

erhält. Ein Lauf ρ eines PBA ist akzeptierend, wenn für jeden Pfad π die höchste unendlich oft auftretende Priorität gerade ist: $\max(\inf(c(\rho(\pi))))$ ist gerade. Ein äquivalenter MBA kann leicht durch

$$\mathcal{F}_c = \{S \subseteq Q | \max(c(S)) \text{ gerade}\}$$

bestimmt werden.

Übersetzung von MBA in PBA

- Grundidee: Wenn man auf einem Pfad bei jedem Zustand q, den man besucht, die Menge S der Zustände weiß, die man seit dem letzten Besuch von q gesehen hat, dann tritt $\inf(\rho(\pi))$ unendlich oft als so ein s auf. Außerdem sind alle diese S, die unendlich oft auftreten, Teilmenge von $\inf(\rho(\pi))$.
- Folie: "Beispiel für $S_{\rho,i}$"

Zu $\rho \in Q^\omega$ bezeichne $\rho(i)$ das i-te Element (beginnend bei 0).

$$S_{\rho,i} = \begin{cases} \emptyset & \text{falls } \rho(j) \neq \rho(i) \forall j < i \\ \{\rho(j+1), \ldots, \rho(i)\} & \text{für das } \max j < i \text{ mit } \rho(j) = \rho(i) \end{cases}$$

Lemma: Ist $\rho \in Q^\omega$, so ist $S_{\rho,i} = \inf(\rho)$ für unendlich viele $i \in \mathbb{N}$ und $S_{\rho,i} \not\subseteq \inf(\rho)$ für nur endlich viele i.

Beweis: Wähle m und m' mit:

- für jedes $i \in \mathbb{N}$ mit $\rho(i) \notin \inf(\rho)$ gilt $i < m$.
- für alle $q \in \inf(\rho)$ existiert ein j mit $m < j < m'$ und $\rho(j) = q$.

Dann ist klar, daß für alle $i > m'$ gilt, daß $S_{\rho,i} \subseteq \inf(\rho)$. Wir zeigen, daß es für jedes $i \geq m'$ ein $j \geq i$ gibt mit $S_{\rho,j} = \inf(\rho)$. Sei dazu q der Zustand aus $\inf(\rho)$, dessen Besuch an der Stelle i am weitesten zurück liegt. Dann wurden seitdem alle anderen Zustände aus $\inf(\rho)$ bereits besucht. Da $q \in \inf(\rho)$ existiert ein $j > i$ mit $\rho(j) = q$. Dann ist $S_{\rho,j} = \inf(\rho)$ für das kleinste dieser j.

q.e.d.

Setze nun für $S \subseteq Q$

$$c_{\mathcal{F}}(S) = \begin{cases} 2 \cdot |S| & \text{falls } S \in \mathcal{F} \\ 2 \cdot |S| - 1 & \text{falls } S \notin \mathcal{F} \end{cases}$$

Lemma: Sei $\rho \in Q^\omega$ und sei n der maximale Wert für $c_{\mathcal{F}}(S_{\rho,i})$, der unendlicht oft auftritt. Dann ist n gerade genau dann, wenn $\inf(\rho) \in \mathcal{F}$.

q.e.d.

Lemma: Ist $\rho \in Q^\omega$ und $LAR(\rho) = \vec{l}_0 \vec{l}_1 \ldots$, dann gilt hitmenge($\vec{l}_i = S_{\rho,i}$ für alle i mit $S_{\rho,i} \neq \emptyset$.

q.e.d.

Satz: Zu jedem MBA $\mathfrak{A} = (Q, \Sigma, Q_{in}, \Delta, \mathcal{F})$ kann man einen äquivalenten PBA konstruieren.

Beweis: Definiere $\mathfrak{A}' = (Q', \Sigma, Q'_{in}, \Delta', c)$ durch

- $Q' = LAR_Q$,
- $Q'_{in} = \{\text{next}(\vec{l}_{in}, q) | q \in Q_{in}\}$
- $\vec{l}, a, \vec{l}_1, \ldots, \vec{l}_k \in \Delta'$,

wenn es $q, a, q_1, \ldots, q_k) \in \Delta$ gibt mit: q ist der erste Zustand in $\vec{l}$ und $\vec{l}_i = \text{next}(\vec{l}, q_i)$.

$c(\vec{l}) = c_{\mathcal{F}}(\text{hitmenge}(\vec{l}))$

Mit den vorherigen Betrachtungen gilt dann $T(\mathfrak{A}) = T(\mathfrak{A}')$.

q.e.d.

[2004/11/30]

Bisher 3 BA-Modelle:

- Büchi BBA, Endzustände F
- Muller MBA, Familie $\mathcal{F}$
- Parität PBA, Funktion c: Lauf ρ akzeptiert $t \Leftrightarrow \forall$ Pfade π von t: $\max(\text{Inf}(c(\rho(\pi))))$ gerade
- PBA und MBA gleiche Ausdrucksstärke (MBA $\rightarrow$ PBA via LAR), BBA echt schwächer
- alle drei BAs sind unter Vereinigung abgeschlossen
- BBA nicht unter Komplement abgeschlossen

Frage: Komplement für MBA / PBA ?

- Nächste Woche: Antwort Ja.
- Vorbereitung: Unendliche Spiele

Spiele / BAs (Idee): Sei $\mathfrak{A} = (Q, \Sigma, Q_{in}, \Delta, c)$ PBA für Binärbäume, t Eingabebaum. Zwei Spieler: A (für Automat), P (Pfad-Sucher)

- A wählt Starttransition, also $q \in Q_{in}$ und $(q, t(\varepsilon), q', q'') \in \Delta$
- P wählt rechts / links, entsprechend wird 1 oder 2 und q' bzw. q'' neue Position
- A wählt wieder kompatible Transition (hier: $(q'', t(2), q''', q^{iv}) \in \Delta$)

Wiederhole dies ad infinitum. Erhalten Folge von Zuständen $\rho = qq''$. Das Spiel wird von Δ gewonnen, falls ρ die Akzeptanzbedingung des Automaten $\mathfrak{A}$ erfüllt, ansonsten gewinnt P.

Bemerkung:

1. Unendliches Spiel
2. A versucht, $t \in T(\mathfrak{A})$ zu zeigen, P versucht, $t \notin T(\mathfrak{A})$ zu zeigen, indem er ein "Gegenbeispiel" konstruiert.
3. Spieler fungieren als intelligente Blackbox, um "globales Wissen" in "lokale Entscheidungen" zu übertragen.
4. $t \in T(\mathfrak{A})$: "Es existiert ein Lauf für alle Pfade, sodaß die Akzeptanzbedingung erfüllt ist."
   $t \notin T(\mathfrak{A})$: "Für alle Läufe existiert ein Pfad, sodaß die Akzeptanzbedingung nicht erfüllt ist."

5 Unendliche Spiele:

(Genauer: Zwei-Personen, Null-Summen-Spiele mit perfekter Information) $\sim$ Analogie zu Schach

Definition (Spiel): $\mathcal{G} = (G, \text{Win})$, Zwei Spieler: 0 und 1

- $G = (Q, Q_0, E)$ ist Spielgraph, wobei Q eine Menge von Knoten (Zustände) ist, $Q_0 \subseteq Q$ sind Knoten von Spieler 0 und $E \subseteq Q \times Q$ sind Knotenrelationen (Transitionen). Wir setzen BA-vollständige Graphen voraus: $\forall q \in Q \exists q' \in Q : (q, q') \in E$, d.h. für jeden Knoten existiert ein Nachfolger. $Q_1 = Q \setminus Q_0$ sind die Knoten von Spieler 1.
- Win ist Gewinnbedingung für Spieler 0
- Spielverlauf ist ω-Folge $rho = q_0 q_1 q_2 \ldots$, mit $(q_i, q_{i+1}) \in E$ und
  - ist $q_i \in Q_0$, dann wählt Spieler 0 den Nachfolger q_{i+1},
  - ist $q_i \in Q_1$, dann wählt Spieler 1 den Nachfolger q_{i+1}.

  Intuition: Schiebe Spielstein entlang der Kanten

  Spielverlauf = Partie

  - Partie ρ: Spieler 0 gewinnt ρ, falls ρ die Bedingung Win erfüllt. [Win kann automatentheoretische Bedingung oder Logik-Formel sein. Formal: Win $\subseteq Q^\omega$, Spieler 0 gewinnt $\rho :\Leftrightarrow \rho \in$ Win]

  Darstellung von G: $\bigcirc$ für Elemente in Q_0, $\Box$ für Q_1.

Definition (Strategie): Eine **Strategie für Spieler 0 von q aus** ist eine Funktion $f_= : Q^+ \to Q$ mit: f_0 gibt für jedes Präfix $q_0 \ldots q_i$ einer Partie mit $q_0 = q$ und $q_i \in Q_0$ ein q_{i+1} mit $(q_i, q_{i+1}) \in E$. [Q^+: Man "merkt" sich den ganzen bisherigen Verlauf.]

- Eine Partie ρ ist gemäß einer Strategie f_0, falls $q_0 = q$ und für alle $q_i \in Q_0$ gilt: $q_{i+1} = f(q_0 \ldots q_i)$.
- f_0 heißt Gewinnstrategie für Spieler 0 von q aus, falls Spieler 0 alle Partien ρ von q aus gewinnt, falls er gemäß f_0 spielt.

Analog alles für Spieler 1:

- Gewinnbereich:

  $w_0 = \{q \in Q | \text{Spieler 0 gewinnt von } q \text{ aus}\}$

  $w_1 = \{q \in Q | \text{Spieler 1 gewinnt von } q \text{ aus}\}$

Bemerkung: Im zweiten Beispiel hängt der Wert der Gewinnstrategie f_0 auf w_0 nicht von w ab für $q \in w_0$.

Definition (positional): Eine Strategie f_0 von Spieler 0 von q aus heißt **positional**, falls $f(qq_1 \ldots q_i)$ für alle $q_i \in W_0$ nur von q_i abhängt. [lokal oder speicherlos]
Analog für Spieler 1.

Bemerkung:

1. Sehr einfach, als Datenstruktur effizient.
2. Können wir schreiben als $F_0 : Q_0 \to Q$
3. Darstellung als Untergraph von G: Für Q_0-Knoten genau einen Nachfolger, und für Q_1-Knoten alle alten Nachfolger

Beispiele: $W_0 \cup W_1 = Q$

- Frage: Gilt das allgemein ?

Definition (determiniert): Ein Spiel heißt **determiniert**, falls $W_0 \cup W_1 = Q$.

Bemerkung: Es gibt (sehr exotische) Speiel, die nicht determiniert sind, alle in der Vorlesung betrachteten Spiele sind aber determiniert.

Ziel: Zeige positionale Determiniertheit für Paritätsspiele. Dazu wird zunächst das Erreichbarkeits- / Garantiespiel betrachtet:

- Graph $G = (Q, Q_0, E), F \subseteq Q$, Q endlich
- Win: Spieler 0 gewinnt gdw. $\exists i : \rho(i) \in F$
- Bestimme den Gewinnbereich W_0: Bilde 0-Attraktor

  $$\mathrm{Attr}_0^i(F) := \{q \in Q | \text{Spieler 0 kann in höchstens } i \text{ Zügen den Besuch von } F \text{ erzwingen}\}$$

  Induktiv:

  - $\mathrm{Attr}_0^0(F) = F$
  - $\mathrm{Attr}_0^{i+1} = \mathrm{Attr}_0^i(F) \cup \{q \in Q_0 | \exists (q,p) \in E : p \in \mathrm{Attr}_0^i(F)\} \cup \{q \in Q_1 | \forall (q,p) \in E : p \in \mathrm{Attr}_0^i(F)\}$

  Offensichtlich gilt $\mathrm{Attr}_0^0(F) \subseteq \mathrm{Attr}_0^1(F) \subseteq \mathrm{Attr}_0^2(F) \subseteq \ldots$. Da Q endlich ist, ist die Folge stationär, also gibt es ein $jleq|Q|$ mit $\mathrm{Attr}_0^j(F) = \mathrm{Attr}_0^{j+1}(F)$. Seien $\mathrm{Attr}_0(F) := \bigcup_{i=0}^{|Q|} \mathrm{Attr}_0^i(F)$ genau die Zustände, von denen aus Spieler 0 einen Besuch von F erzwingen kann.
  Attr-Strategie: Verringere Abstand zu F, positionale Strategie, Details nochmal in anderer Vorlesung.
- Bemerkung: Positionale Strategien können noch vom Anfangszustand abhängen.

Definition (uniform): Eine Strategie f_0 von Spieler 0 heißt **uniform**, falls f_0 eine Gewinnstrategie von q aus für jedes $q \in W_0$ ist.

Lemma: Für Paritätsspiele gilt: Falls eine positionale Strategie für jedes $q \in W_0$ existiert, dann existiert eine uniforme positionale Gewinnstrategie.

Beweis: Übung.

Satz: Paritätsspiele sind positional determiniert, d.h. $Q = W_0 \cup W_1$ und jeder Spieler hat auf seinem Gewinnbereich eine positionale Gewinnstrategie.

Beweis: Sei $G = (Q, Q_0, E)$, $c : Q \to \{0, \ldots, k\}$. Sei $c_k = \{q \in Q | c(q) = k\}$. Induktion über k:

- $k = 0 \Rightarrow 0$ gerade, also $W_0 = Q$ und jede Strategie ist eine Gewinnstrategie.
- Induktionsschritt: Sei k die höchste Priorität und O.B.d.A. k gerade (andernfalls: Vertausche Spieler 0/1 in der Partie). Sei $P_1 \subseteq Q$ die Menge der Zustände, sodaß Spieler 1 eine positionale Gewinnstrategie besitzt.

Zeige: $W_0 = Q \setminus P_1$ und Spieler 0 kann positional gewinnen. Dann gilt: $W_1 = P_1, W_0 \cup W_1 = Q$.
Sei f_1 eine uniforme positionale Gewinnstrategie auf P_1 für Spieler 1.

1. Fall: $Q \setminus P_1 \cap C_k = \emptyset$
   $Q \setminus P_1$ definiert ein Unterspiel, es existiert also nur eine Priorität kleiner als k. Nach Induktionsvoraussetzung gibt es eine Partition in W_0', W_1' mit positionaler Gewinnstrategie.

[2004/12/07]

Satz: Paritätsspiele sind determiniert, und beide Spieler besitzen positionale Gewinnstrategien auf ihren Gewinnregionen.

Beweis: Induktion über die Anzahl der Farben; o.B.d.A. sei die höchste Farbe gerade.

1. Fall: $Q \setminus P_1$ [P_1 Gewinnregion von Spieler 1 mit positionaler Gewinnstrategie] enthält einen Knoten höchster Farbe.
2. Fall: $Q \setminus P_1$ enthält Knoten der höchsten Farbe k (gerade). Dabei gilt $\mathrm{Attr}_0(C_k \setminus P_1) \cap P_1 = \emptyset$. $(Q \setminus P_1) \setminus \mathrm{Attr}_0(C_k \setminus P_1)$ ist der Graph eines Subspiels mit Farbmenge $\subseteq \{1, \ldots, k-1\}$. Die Induktionsvoraussetzung liefert eine Zerlegung in die Gewinnregionen W_0', W_1' (jeweils mit positionaler Gewinnstrategie für 0 bzw. 1).

Beachte: $W_1' = \emptyset$, da jeder Knoten im Gesamtspiel zur Gewinnregion von Spieler 1 gehört.

Strategie für Spieler 0

a) Auf W_0' existiert eine Gewinnstrategie gemäß Induktionsvoraussetzung.
b) Im Attraktor existiert eine Gewinnstrategie auf $\mathrm{Attr}_0(C_k \setminus P_1)$.

Also existiert eine Gewinnstrategie für Spieler 0.

Übergang zu Baumautomaten Zu einem Paritätsbaumautomat $\mathfrak{A} = (Q, \Sigma, Q_{in}, \Delta, c)$ und einem Eingabebaum t führe einen Spielgraphen $G_{\mathfrak{A},t}$ (und ein entsprechendes Paritätsspiel) ein:

- Spielposition von Spieler 0 (Automat): Position x, Buchstabe $t(x)$, Zustand bei x
  $Q_0 = \{0,1\}^* \times \Sigma \times Q$, $[2] = \{0,1\}$
- Spielposition von Spieler 1 (Pfadfinder): $Q_1 = \{0,1\}^* \times \Sigma \times \Delta$
- Transitionen:
  - $(x, t(x), q) \to (x, t(x), \underbrace{(q, t(x), q', q'')}_{\in \Delta})$
  - $(x, t(x), (q, t, q', q'')) \begin{array}{ll} \nearrow & (x0, t(x0), q') \\ \searrow & (x1, t(x1), q'') \end{array}$
  - $c((x, t(x), q)) := c(q)$
  - $c((x, t(x), (q, t(x), q', q'')) := c(q)$
- Gewinnbedingung: Paritätsbedingung

Lemma: $\mathfrak{A}$ akzeptiert $t \Leftrightarrow$ Im Paritätsspiel über $G_{\mathfrak{A},t}$ hat Spieler Automat von einer Position $(\varepsilon, t(\varepsilon), q)$ mit $q \in Q_{in}$ eine positionale Gewinnstrategie.

Beweis:

- $\Rightarrow$) Nach Voraussetzung wähle einen akzeptierenden Lauf: Ein Lauf liefert eine Strategie für einen Automat. Nach der Definition der Färbung ergibt sich sogar eine Gewinnstrategie.
- $\Leftarrow$) Eine Gewinnstrategie ist gegeben. Die Verfolgung für alle Wahlen eines Pfadfinders liefert einen Lauf. Nach der Definition ist dieser Lauf akzeptierend.

Komplementsatz (für PBA): Zu einem PBA $\mathfrak{A}$ kann man einen PBA $\mathfrak{B}$ konstruieren mit $T(\mathfrak{B}) = T_\Sigma^\omega \setminus T(\mathfrak{A})$.

Beweis:

- Aufgabe: Zu $\mathfrak{A}$ konstruiere $\mathfrak{B}$ mit $\mathfrak{A}$ akzeptiert t nicht $\Leftrightarrow$ $\mathfrak{B}$ akzeptiert t.
- Beachte: $\mathfrak{A}$ akzeptiert t nicht $\Leftrightarrow$ In einem Paritätsspiel über $G_{\mathfrak{A},t}$ hat der Automat keine positionale Gewinnstrategie $\underset{\text{det.}}{\Leftrightarrow}$ In einem Paritätsspiel über $G_{\mathfrak{A},t}$ hat der Pfadfinder eine positionale Gewinnstrategie.
- Pfadfinder-Strategie:

  $$f : \{0,1\}^* \times \Sigma \times \Delta \to \{0,1\} \qquad (x, t(x), \tau(x)) \mapsto \{0,1\}$$

  Die Parametrisierung führt auf Funktionen $f_x : \Sigma \times \Delta \to \{0,1\}$. Die Menge aller möglichen f_x ist endlich:

  $$I = \{g | g : \Sigma \times \Delta \to \{0,1\}\}$$

  Eine Strategie vom Pfadfinder ist kodierbar durch die Abbildung

  $$s : x \mapsto \text{passendes } f_x \qquad s : \{0,1\}^* \to I$$

  s ist ein mit I-Werten beschrifteter Baum ($s \in T_I^\omega$).
- Idee für $\mathfrak{B}$: Rate auf t den Strategiebaum s für den Pfadfinder und überprüfe, daß eine Gewinnstrategie für den Pfadfinder vorliegt.
- Der Pfadfinder gewinnt über $G_{\mathfrak{A},t}$ *gdw* Es existiert ein I-beschrifteter Baum, sodaß für alle Transitionsfolgen $\tau_0, \tau_1, \tau_2, \ldots$ durch den Automat und für alle Pfade π, die durch $\tau_0, \tau_1, \ldots$ und s induziert sind, gilt: Die Paritätsbedingung wird verletzt. $\Leftrightarrow$ Es existiert ein I-beschrifteter Baum s, sodaß für $t\hat{s}$ [1] gilt: Für alle Pfade π und für alle Transitionsfolgen $\tau_0\tau_1\tau_2\ldots$ gilt: Wenn π gemäß s für $\tau_0\tau_1\ldots$ gewählt wird, dann ist die Paritätsbedingung verletzt. [2]

[2004/12/20]

Anmerkung: Der vorgestellte Beweis zur positionalen Determiniertheit funktioniert nur für endliche Graphen. Da die Induktion über die Anzahl der Prioritäten läuft, kann der Beweis für unendliche Graphenmit endlich vielen Prioritäten angepaßt werden.

[1] $t\hat{s}$ ist eine Beschriftung in $\Sigma \times I$

[2] Folge über $I \times \Sigma \times \{0,1\} \times \Delta$ – testbar durch Muller-Automaten

6 Leerheitstest für Paritätsbaumautomaten

Zur Vereinfachung betrachten wir 2-verzweigte Bäume. Ab jetzt ist wieder $[2] = \{1, 2\}$.
Das Leerheitsproblem für PBAs ist das folgende Entscheidungsproblem:

- Gegeben: PBA $\mathfrak{A}$
- Frage: Gilt $T(\mathfrak{A}) = \emptyset$?

Wir werden diese Frage auf die Existenz einer Gewinnstrategie in einem Paritätsspiel $(G_{\mathfrak{A}}, c_{\mathfrak{A}})$ über einem **endlichen** Spielgraphen $G_{\mathfrak{A}}$ reduzieren.

Idee: In dem Spiel $G_{\mathfrak{A},t}$ ist ein Baum t gegeben und Spieler 0 versucht zu zeigen, daß $t \in T(\mathfrak{A})$ ist. Bei dem Leerheitsproblem geht es nur um die Existenz eines Baumes in $T(\mathfrak{A})$. Dieser wird von Spieler 0 in $G_{\mathfrak{A}}$ während des Spielverlaufs aufgebaut.
Ab jetzt werden Knotenmengen in Spielgraphen mit V, V_0 und V_1 bezeichnet.

Definition von $(G_{\mathfrak{A}}, c_{sA})$**:** Sei $\mathfrak{A} = (Q, \Sigma, Q_{in}, \Delta, c)$ mit

- $G_{\mathfrak{A}} = (V, V_0, E)$ mit $V = Q \cup \Delta$, $V_0 = Q$ und $V_1 = \Delta$ und
  - für alle $q \in Q$ und alle $(q, a, q_1, q_2) \in \Delta$ ist eine Kante von $q \to (q, a, q_1, q_2) \in E$
  - für alle $(q, a, q_1, q_2) \in \Delta$) sind die Kanten $(q, a, q_1, q_2) \to q_1$ und $(q, a, q_1, q_2) \to q_2$ in E.
- $c_{\mathfrak{A}}(q) = c(q)$ und $c_{\mathfrak{A}}((q, a, q_1, q_2)) = 0$.

Satz: Sei $\mathfrak{A} = (Q, \Sigma, Q_{\in}\Delta, c)$. Dann gilt $T_{\mathfrak{A}} \neq \emptyset$ genau dann, wenn Spieler 0 in $(G_{\mathfrak{A}}, c_{\mathfrak{A}})$ von einem Zustand $q_0 \in Q_{\in}$ eine Gewinnstrategie hat.

Beweis:

- $\Rightarrow$: Sei $t \in T(\mathfrak{A})$ und ρ ein akzeptierender Lauf von $\mathfrak{A}$ auf t. Setze $q = \rho(t)$. Eine Gewinnstrategie f_0 läßt sich wie folgt definieren:
  Sei $w = q_0(q_0, a_0, q_1^1, q_1^2)q_1^{i_1}(q_1^{i_1}, a_1, q_2^1, q_2^2)q_2^{i_2} \dots (q_{n-1}, a_{n-1}, q_n^1, q_n^2)q_n^{i_n}$ ein nach f_0 gespielter Partieanfang mit $n \geq 0$ und $i_j \in [2]$. Setze $x = i_1, \dots, i_n$. Definiere $f_0(\omega) = (q_n^{i_n}, t(x), \rho(x1), \rho(x2))$. Per Induktion über n läßt sich leicht zeigen, daß $q_n^{i_n} = \rho(x)$ und somit ist $f_0(w) \in \Delta$ eine korrekte Wahl für Spieler 0. Die Zustandsfolge einer nach f_0 gespielten Partie entspricht einem Pfad durch ρ und erfüllt somit die Paritätsbedingung, da ρ akzeptierend ist. Also ist f_0 eine Gewinnstrategie.
- $\Leftarrow$: Sei f_0 eine Gewinnstrategie für Spieler 0 von $q_0 \in Q_{in}$ aus. Da Paritätsspiele positional determiniert sind, können wir f_0 positional wählen. Wir definieren simultan einen Baum t und einen akzeptierenden Lauf ρ von $\mathfrak{A}$ auf t. $\rho(\varepsilon) := q_0$. Sei $x = x'i$ mit $x' \in [2]^*, i \in [2]$. Sei weiterhin $q = \rho(x')$ und $(q, a, q_1, q_2) = f_0(q)$. Setze $t(x') := a$ und $\rho(x) = q_1$. Dann ist ρ ein Lauf von $\mathfrak{A}$ auf t. Nach der Definition von ρ entspricht jeder Pfad durch ρ einer Zustandsfolge einer nach f_0 gespielten Partie und erfüllt somit die Paritätsbedingung. Also ist ρ akzeptierend und $t \in T(\mathfrak{A})$.

q.e.d.

Satz: Das Problem

- Gegeben: Endliches Paritätsspiel (G, c) und ein Knoten v von G.
- Frage: Hat Spieler 0 eine Gewinnstrategie von v aus ?

ist in NP (also insbesondere entscheidbar).

Beweis: Wenn Spieler 0 eine Gewinnstrategie von v aus hat, dann auch eine positionale. Jede positionale Strategie läßt sich als Teilmenge der Kantenmenge von G darstellen, die zu jedem Spieler-0-Knoten genau einen Nachfolger auswählt. So eine Strategie kann nichtdeterministisch geraten werden. Für den resultierenden Spielgraphen kann in Polynomzeit verifiziert werden, daß alle Pfade die Paritätsbedingung erfüllen.

q.e.d.

Satz: Das Leerheitsproblem für PBAs ist in NP.

Beweis: Das Spiel $(G_{\mathfrak{A}}, c_{\mathfrak{A}})$ kann in Polynomzeit aus $\mathfrak{A}$ konstruiert werden und q_0 kann nichtdeterministisch geraten werden.

q.e.d.

Reguläre Bäume: Ein Baum $t \in T_\Sigma^\omega$ heißt regulär, wenn er von einem DEA mit Ausgabe erzeugt werden kann. Sei ein DEA hat die Form $\mathfrak{B} = (Q_{\mathfrak{B}}, [2], \Sigma, q_{in,\mathfrak{B}}, \delta_{\mathfrak{B}}, \lambda_{sB})$ mit

- Zustandsmenge $q_{\mathfrak{B}}$,
- Eingabealphabet [2],
- Ausgabealphabet Σ,
- Transitionsfunktion $\delta_{\mathfrak{B}}$ und
- Ausgabefunktion $\lambda_B : Q \to \Sigma$.

Wie üblich erweitern wir $\delta_{\mathfrak{B}}$ auf Eingabewörter: $\delta_{\mathfrak{B}}(q, \varepsilon) = q$ und $\delta_{\mathfrak{B}}(q, xi) = \delta_{\mathfrak{B}}(\delta_{\mathfrak{B}}(q, x), i)$. Der von $\mathfrak{B}$ generierte Baum $t_{\mathfrak{B}}$ ist dann definiert durch

$$t_{\mathfrak{B}}(x) = \lambda_{\mathfrak{B}}(\delta_{\mathfrak{B}}(q_{in,\mathfrak{B}}, x)).$$

Satz: Ist $\mathfrak{A}$ ein PBA, dann enthält $T(\mathfrak{A})$ einen regulären Baum, wenn $T(\mathfrak{A}) \neq \emptyset$ gilt.

Beweis: Der Baum von $T(\mathfrak{A})$, der aus einer positionalen Strategie f_0 für Spieler 0 in $(G_{\mathfrak{A}}, c_{\mathfrak{A}})$ von $q_0 \in Q_\in$ konstruiert wurde, ist regulär. (Beweis: Übung)
Der Automat $\mathfrak{B}$ für die positionale Strategie aus dem Beispiel $Q_{\mathfrak{B}} = \{q_0, q_1, q_2\}, q_{in,\mathfrak{B}} = q_0$

- $\lambda_{\mathfrak{B}}(q_0) = b$
- $\lambda_{\mathfrak{B}}(q_1) = b$
- $\lambda_{\mathfrak{B}}(q_2) = a$

7 S2S und der Satz von Rabin

[2004/12/21]

Die Logik S2S (second-order system of 2 successors) ist definiert über Variablen $x, y, \ldots$ erster Stufe interpretiert als einzelne Baumknoten und Variablen $X, Y, \ldots$ zweiter Stufe, die als Mangen von Baumknoten interpretiert werden.

S2S-Terme:

- ε ist ein S2S-Term
- Jede Variable x erster Stufe ist ein S2S-Term.
- Ist τ ein S2S-Term, dann sind $\tau 1$ und $\tau 2$ S2S-Terme.

S2S-Formeln:

- Sind σ und τ Terme, dann sind $X(\sigma)$ und $\sigma = \tau$ S2S-Formeln (für zweite Stufe Variable X).
- Sind φ_1 und φ_2 S2S-Formeln, dann sind $\varphi_1 \vee \varphi_2$, $\neg\varphi_1$, $\exists x \varphi_1$ und $\exists X \varphi_1$ S2S-Formeln. Wie üblich ist $\forall x \varphi = \neg\exists x \neg\varphi$.

Die Struktur, über der wir diese Formeln betrachten, ist der unbeschriftete binäre Baum $\underline{T_2} = ([2]^*, \varepsilon, S1, S2)$, wobei $S1$ und $S2$ als die entsprechenden Nachfolgerfunktionen interpretiert werden: $S_1(u) = u1, S_2(u) = u2$.
Die **S2S-Theorie** ist die Menge als S2S-Sätze (Formeln ohne freie Variablen), die in $\underline{T_2}$ wahr sind. Diese Theorie wird auch monadische Theorie zweiter Stufe genannt und mit $MTh_2(T_2)$ bezeichnet.

Beispiele:

- $X \subseteq Y : \forall z(X(z) \to Y(z))$
- $X = Y : X \subseteq Y \wedge Y \subseteq X$
- Chain(X): Die Elemente in X sind linear geordnet durch $\sqsubseteq$: $\forall x, y((X(x) \wedge X(y)) \to (x \sqsubseteq y \vee y \sqsubseteq x)$
- $X \sqsubseteq Y$: Jede Menge, die y enthält und unter Vorgängern abgeschlossen ist, enthält auch x: $\forall X((X(y) \wedge \forall z([X(z1) \vee X(z2)] \to X(z))) \to X(x))$
- Path(X): X ist ein Pfad, also eine maximale Kette: Chain$(X) \wedge \neg\exists Y(\text{Chain}(Y) \wedge X \subseteq Y \wedge \neg(X = Y))$

Modellbeziehung von S2S-Formeln: S2S-Formeln mit freien Mengenvariablen $\varphi(X_1, \ldots, X_n)$ werden in erweiterten Strukturen $\underline{t} = (\underline{T_2}, P_1, \ldots, P_n)$ mit (einstelligen) Prädikaten P_i, die als Interpretation der X_i dienen, betrachtet. Wir schreiben $\underline{t} \vDash \varphi(X_1, \ldots, X_n)$, wenn φ in $\underline{t}$ gilt (mit der natürlichen Semantik). $\underline{t}$ wird mit dem Baum $t \in T^\omega_{\mathbb{B}^n}(\mathbb{B} = \{0, 1\})$ identifiziert, der durch $t(x) = (b_1, \ldots, b_n)$ mit $b_i = 1$, falls $x \in P_i$ und $b_i = 0$, falls $x \notin P_i$ definiert ist.
Eine S2S-Formel $\varphi(X_1, \ldots, X_n)$ definiert die Baumsprache $T(\varphi) = \{t \in T^\omega_{\mathbb{B}^n} | \underline{t} \vDash \varphi\}$.

Bemerkung: Für $n = 0$ ist $\mathbb{B}^n$ ein einelementiges Alphabet.

Beispiel für $n = 1$: $T = \{t \in T^\omega_{\mathbb{B}} | t \text{ enthält einen Pfad mit unendlich vielen } 0\}$

$$\varphi(X_1) = \exists Y(\text{Path}(Y) \wedge \forall x \in Y \exists y \in Y(x \sqsubseteq y \wedge \neg X_1(y)))$$

Dann ist $T(\varphi) = T$.

Lemma: Ist $T \subseteq T^\omega_{\mathbb{B}^n}$ PBA-erkennbar, so ist $T = T(\varphi)$ für eine S2S-Formel $\varphi(X_1, \ldots, X_n)$.

Beweis: Sei $T = T(\mathfrak{A})$ für einen PBA $\mathfrak{A} = (Q, \mathbb{B}^n, Q_{in}, \Delta, c)$ mit $Q = \{1, \ldots, l\}$ und $c : Q \to \{0, \ldots, m\}$.

- Idee: Beschreibe einen akzeptierenden Lauf in S2S. Die Menge Y_i enthält die Baumknoten, die in dem Lauf mit dem Zustand i beschriftet werden.
- Hilfsformel:
  - $\text{Partition}(Y_1, \ldots, X_l) = \forall x([\bigvee_{i \in [l]} Y_i(x)] \wedge [\bigwedge_{i,j \in [l], i \neq j} (Y_i(x) \to \neg Y_j(x))])$

Für $a \in \mathbb{B}^n, a = (b_1, \ldots, b_n)$ sei $\psi_a = (b_1)X_1(x) \wedge \ldots \wedge (b_n)X_n(x)$, $(b_i) = \neg$ falls $b_i = 0$ und leer sonst.

$$\begin{aligned}\varphi(X_1, \ldots, X_n) = \quad & \exists(Y_1, \ldots, Y_l)(\text{Partition}(Y_1, \ldots, Y_l) \wedge (\bigvee_{i \in Q_{in}} Y_i(\varepsilon)) \\ & \wedge \forall x(\bigvee_{(i,a,i_1,i_2) \in \Delta} (Y_i(x) \wedge Y_{i_1}(x1) \wedge Y_{i_2}(x2) \wedge \psi_a(x))) \\ & \wedge \forall z(\text{Path}(z) \to (\bigvee_{i \in Q, c(i) \text{ gerade}} ([\forall x \in Z \exists y \in Z(x \sqsubseteq y \wedge Y_i(y))] \\ & \wedge \bigwedge_{j \in Q, c(j) > c(i)} \exists x \in Z \forall y \in Z(x \sqsubseteq y \to \neg X_j(y))))\end{aligned}$$

Dann gilt $T = T(\varphi)$.

q.e.d.

$S2S_0$**:**

- Nur Mengenvariablen.
- Atomare Formeln:
  - $X \subseteq Y$
  - $\text{Sing}(X)$ (X ist einelementig)
  - $S_1(X, Y)$ (X, Y sind einelementig und das Element aus Y ist der 1-Nachfolger des Elements aus X)
  - $S_2(X, Y)$ analog
- Komplexere Formeln durch Boolesche Operatoren und Quantoren (für Mengenvariablen).

Bemerkung: S2S und $S2S_0$ haben die gleiche Ausdrucksstärke.

Bemerkung: Die Klasse der PBA-erkennbaren Sprachen ist unter Vereinigung abgeschlossen.

Projektion: Seien Σ_1 und Σ_2 Alphabete und $h : \Sigma_1 \to \Sigma_2$. Zu $t \in T^\omega_{\Sigma_1}$ sei $h(t) \in T^\omega_{\Sigma_2}$ definiert durch $h(t)(x) = h(t(x))$ und für $T \subseteq T^\omega_{\Sigma_1}$ sei $h(T) \subseteq T^\omega_{\Sigma_2}$ definiert durch

$$h(T) = \{t \in T^\omega_{\Sigma_2} | t = h(t') \text{ mit } t' \in T\}$$

Projektionslemma: Die Klasse der PBA-erkennbaren Baumsprachen ist unter Projektion abgeschlossen, d.h. ist $T \subseteq T^\omega_{\Sigma_1}$ PBA-erkennbar, so auch $h(T)$.

Beweis: Ersetze die Transitionen (q, a, q_1, q_2) in einem PBA für T durch $(q, h(a), q_1, q_2)$.

q.e.d.

Lemma: Zu jeder S2S-Formel $\varphi(X_1, \ldots, X_n)$ kann man einen PBA $\mathfrak{A}$ konstruieren mit $T(\varphi) = T(\mathfrak{A})$.

Beweis: Es reicht, das Ergebnis für $S2S_0$ zu zeigen. Wie gehen induktiv über den Formelaufbau vor:

- Induktionsanfang (atomare Formeln):
  - $X \subseteq Y$: Baumautomat mit einen Zustand und Transitionen $\binom{X}{Y} \to \binom{0}{0}, \binom{0}{1}, \binom{1}{1}$, $c(q_0) = 0$.

    [Grafik]
  - $\mathrm{Sing}(X)$:

    [Grafik]
  - $S_1(X, Y)$:

    [Grafik]
  - $S_2(X, Y)$: analog
- Induktionsschritt:
  - Negation: Komplement
  - Disjunktion: Vereinigung
  - Existentielle Quantifizierung: Projektion $\exists x_n(\varphi(X_1, \ldots, X_n))$ – Automat für φ über $\mathbb{B}^n$, für ψ über $\mathbb{B}^{n-1}$. $h : \mathbb{B}^n \to \mathbb{B}^{n-1}$ definiert durch $h(b_1, \ldots, b_n) = (b_1, \ldots, b_{n-1})$ für Projektionslemma.

q.e.d.

Satz von Rabin: $MTh(\underline{T_2})$ ist entscheidbar.

Beweis: Ein gegebener S2S-Satz φ kann in einen äquivalenten PBA $\mathfrak{A}$ transformiert werden. $\mathfrak{A}$ akzeptiert den unbeschrifteten Baum genau dann, wenn φ in $\underline{T_2}$ gilt. Dies kann mit dem Leerheitstest für PBAs getestet werden.

q.e.d.

Bemerkung: Die Betrachtungen können leicht auf k-verzweigte Bäume und die entsprechende Logik SkS übertragen werden.

[2005/01/11]

Leerheitstest für PBA → Gewinnstrategien in Paritätsspielen

S2S-Formeln $\varphi(X_1, \ldots, X_n) \rightsquigarrow$ PBA über $\{0, 1\}^n$

Algorithmus: Marcin Jurdzinski, 2000 Wir interessieren uns nur für positionale Strategien für Spieler 0. Mit Strategie ist immer so eine Strategie gemeint (auch auf dem aktuellen Übungszetttel). Sei (G, c) mit $G = (V, V_0, E)$ und $c : V \to \{1, \ldots, 2m\}$ ein Paritätsspiel. Für eine Priorität $p \in \{1, \ldots, 2m\}$ sei n_p die Anzahl der Knoten mit Priorität p, also $n_p = |\{v \in V | c(v) = p\}|$. Für eine Strategie f sei

- W_0^f die Menge der Knoten, von denen aus Spieler 0 mit f gewinnt und
- P_f^v die Menge der gemäß f gespielten Partien, die im Knoten v starten.

Bewertungen von Partien:

- Sei $\rho \in V^\omega$ eine von Spieler 0 gewonnene Partie und p eine ungerade Parität.
- $\beta_p(\rho)$ sie die Anzahl der Knoten mit Priorität p, die in ρ besucht werden, ohne daß ein Knoten höherer Priorität besucht wurde.
- $\beta(\rho) = (\beta_1(\rho), \beta_3(\rho), \ldots, \beta_{2m-1}(\rho))$

$(2346)^\omega = 2346(2346)^\omega$

Die Ordnung auf Partiebewertungen entspricht der "umgedrehten lexikographischen Ordnung" (lese Tupel von hinten nach vorne):

$(l_1, \ldots, l_m) \leq (l'_1, \ldots, l'_m) \Leftrightarrow \forall i \in \{1, \ldots, m\}(l'_i < l_i \Rightarrow \exists j > i(l_j < l'_j)$

Für $p \in \{1, \ldots, 2m\}$ und $r = \lfloor \frac{p}{2} \rfloor + 1$ definieren wir $(l_1, \ldots, l_m) \leq p(l'_1, \ldots, l'_m) :\Leftrightarrow (l_1, \ldots, l_m) \leq (l'_r, \ldots, l'_m)$. Definiere die Abbildung $\text{pred}_p : \mathbb{N}^m \to \mathbb{N}^m$ durch

$$\text{pred}_p(l_1, \ldots, l_m) = (l'_1, \ldots, l'_m) \text{ mit } \begin{cases} l'_i = 0 & \text{falls } 2i-1 < p \\ l'_i = l_i + 1 & \text{falls } 2i-1 = p \\ l'_i = l & \text{falls } 2i-1 > p \end{cases}$$

Bemerkungen:

1. Sind ρ und ρ' Partien mit $\rho = v\rho'$ und ist $p = c(v)$, dann gilt $\beta(\rho) = \text{pred}_p(\rho')$.
2. pred_p ist monoton, für $(l1, \ldots, l_m) \leq (l'_1, \ldots, l'_m)$ gilt $\text{pred}_p(l_1, \ldots, l_m) \leq \text{pred}_p(l'_1, \ldots, l'_m)$.
3. Für alle p gilt $(l_1, \ldots, l_m) \leq_p (l_1, \ldots, l_m)$ und für ungerade p $(l_1, \ldots, l_m <_p \text{pred}_p(l_1, \ldots, l_m)$.

Lemma: Ist f eine Strategie, $v \in W_0^f$, $\rho \in P_f^v$ und $p \in \{1, \ldots, 2m\}$ eine ungerade Priorität, dann gilt $\beta_p(\rho) \leq n_p$.

Beweis: Ist $\beta_p(\rho) > n_p$, dann kommt in ρ ein Knoten mit Priorität zweimal vor, ohne daß zwischendurch ein Knoten höherer Priorität gesehen wird. Da f positional ist, kann Spieler 1 dieses Segment unendlich oft wiederholen und so eine Partie in P_f^v erzeugen, in der die höchste vorkommende Priorität p - also ungerade - ist. Dies widerspricht der Wahl von $v \in W_0^f$.

q.e.d.

Die Bewertungen, die uns interessieren, sind aus

$M_G = \{0, \ldots, n_1\} \times \{0, \ldots, n_3\} \times \ldots, \times \{0, \ldots, n_{2n-1}\}$.

Setze $M_G^\infty = M_G \cup \{\infty\}$. Das Maß für eine Strategie f ist eine Abbildung $\mu_f : V \to M_G^\infty$, definiert durch

$$\mu_f(v) = \begin{cases} \infty & v \in W_0^f \\ \max\{\beta(\rho) | \rho \in P_f^v\} & \text{sonst} \end{cases}$$

Setze zusätzlich $(l_1, \ldots, l_m) \leq \infty$ für alle $(l_1, \ldots, l_m) \in \mathbb{N}^m$. Für zwei Abbildungen $\mu, \mu' : V \to M_G^\infty$ schreiben wir $\mu \leq \mu'$, falls $\mu(v) \leq \mu'(v)$ für alle $v \in V$ gilt.

Lemma: Für jede Strategie f, jedes $v_0 \in V$ und jede Partie $\rho = v_0 v_1 \ldots$ aus $P_v^{v_0}$ mit $\beta(\rho) = \mu_f(v)$ gilt $\mu_f(v_0) = \text{pred}_{c(v_0)}(\mu_f(v_1))$.

Lemma: Für jede Strategie f und jedes $v \in V$ mit $p = c(v)$ gilt:

- $\mu_f(v) = \text{pred}_p(\max\{\mu_f(w) | (v, w) \in V\})$ falls $v \in V_1$,
- $\mu_f(v) = \text{pred}_p(\min\{\mu_f(w) | (v, w) \in E\})$ falls $v \in V_0$.

[2004/01/18]

Wie nennen eine Abbildung $\mu : V \to M_G^\infty$ ein Strategiemaß, wenn gilt:

- $\mu(v) \geq \text{pred}_{c(v)}(\max\{\mu(w) | (v, w) \in E\})$, falls $v \in V_1$ und
- $\mu(v) \geq \text{pred}_{c(v)}(|\ \{\mu(w) | (v, w) \in E\})$, falls $v \in V_0$.

Wir setzen $W_\mu = \{v \in V | \mu(v) \neq \infty\}$. Die Strategie f_μ auf W_μ wählt zu jedem $v \in V_0$ ein $u \in V$ mit (v, u) und $\mu(u)) \min\{\mu(w) | (v, w) \in W\}$.

[Folie: "Ein Strategiemaß"]

Lemma: Ist μ ein Strategiemaß, so gilt $\mu_{f_\mu} \leq \mu$. Insbesondere ist f_μ eine Gewinnstrategie auf W_μ.

Beweis (nur zweiter Teil): Zeige $\mu_{f_\mu}(v) \neq \infty$ für alle $v \in W_\mu$. Sei $v_0 \in W_\mu$ und $\rho = v_0 v_1 v_2 \ldots$ in $P_{f_\mu}^{v_0}$ und p die größte Priorität, die unendlic oft in ρ vorkommt. Dann existieren $j_1 < j_2$ mit $c(v_{j_1}) = p$, $v_{j_2} = v_{j_1}$ und $c(v_j) \leq p$ für alle $j_1 \leq j \leq j_2$.
Angenommen, p ist ungerade. Dann erhalten wir $\mu(v_{j_1}) \geq \text{pred}_p(\mu(v_{j_{1+1}}) >_p \mu(v_{j_{1+1}}) \geq_p \mu(v_{j_{1+2}}) \geq_p \ldots \geq_p \mu(v_{j_2}) = \mu(v_{j_1})$. Wir erhalten $\mu(v_{j_1}) >_p \mu(v_{j_1})$. Widerspruch. Also wir ρ von Spieler 0 gewonnen.

[Folie: "Der Operator lift"]

Lemma:

1. Sind $\mu, \mu' : V \to M^\infty$ mit $\mu \leq \mu'$, dann gilt $\text{lift}(\mu, v) \leq \text{lift}(\mu', v)$ für alle $v \in V$.
2. Für jede Strategie f und jedes $v \in V$ gilt $\text{lift}(\mu, v) \leq \mu_f(v)$.

Beweis:

1. Folgt aus der Monotonie von pred_p.
2. Für $v \in V_1$ ist nach Definition von lift

$$\text{lift}(\mu_f, v) = \min\{x \in M_G^\infty | x \geq \text{pred}_{c(v)}(\max\{\mu_f(w) | (v,w) \in E\})\}$$

Da $\mu_f(v) \geq \text{pred}_{c(v)}(\max\{\mu_f(w) | (v,w) \in E\})$ gilt, $\text{lift}(\mu_f, v) \leq \mu_f(v)$. Analog, falls $v \in V_0$.

q.e.d.

[Folie: "Algorithmus: Strategiemaß"]

Satz: Der Algorithmus Strategiemaß terminiert und die im Algorithmus berechnete Strategie ist eine Gewinnstrategie für Spieler 0 auf W_0.

Beweis: Bezeichne die Abbildung μ nach der i-ten Iteration der Schleife mit μ. Es gilt $\mu_0 < \mu_1 < \mu_2 < \ldots$. Da es nur endlich viele Abbildungen von V nach M_G^∞ gibt, terminiert der Algorithmus.
Wir wissen, daß f_μ eine Gewinnstrategie auf W_μ ist. Zeige also $W_\mu = W_0$. Zeige dazu, daß für jede Strategie f gilt: $\mu_i \leq \mu_f$ (für alle i).
Per Induktion über i: Für $i = 0$ ist $\mu(v) = (0, \ldots, 0) \leq \mu_f(v)$. Weiter gilt

$$\begin{array}{lll} \mu_i(v) & \leq \text{lift}(\mu_{i-1}, v) & \text{(entweder ist } \mu_i(v) = \text{lift}(\mu_{i-1}, v) \text{ oder } \mu_i(v) = \mu_{i-1}(v) \leq \text{lift}(\mu_{i-1}, v)) \\ & \leq \text{lift}(\mu_f, v) & \text{(IV und (1) aus Lemma)} \\ & \leq \mu_f(v) & \text{((2) aus Lemma)} \end{array}$$

Ist f eine Gewinnstrategie auf W_0, dann gilt $\mu(v) \leq \mu_f(v) < \infty$ für alle $v \in W_0$. Somit ist $W_\mu = W_0$.

q.e.d.

Komplexität: Die Zuweisung in der Schleife braucht Zeit m. Die Bedingung in der Schleife kann in Zeit $O(|E| \cdot m)$ durchgeführt werden. Auf jeden Knoten kann maximal $|M_G^\infty|$ oft lift angewendet werden. Es gilt

$$\begin{array}{ll} |M_G| & = \prod_{i=1}^{m} (n_{2i-1} + 1) \quad \text{, da } (\sum_{i=1}^{m} (n_{2i-1} + 1) \leq |V|) \\ & \leq \prod_{i=1}^{m} \frac{|V|}{m} \\ & = \left(\frac{|V|}{m}\right)^m \end{array}$$

Der Algorithmus läuft somit in Zeit $O\left(|E| \cdot m \cdot |V| \cdot \left(\frac{|V|}{m}\right)^m\right)$ und in Platz $O(|V| \cdot m)$.

8 Synthese reaktiver Programme

[2005/01/25]

Aufgabe: Entwicklung eines Programmes f, das durch eine Eingabevariable x und eine Ausgabevariable y mit seiner Umgebung kommuniziert. Die Variable x wird von f gelesen und der Umgebung geschrieben, die Variable y umgekehrt von f geschrieben und von der Umgebung gelesen. *f transformiert eine ω-Folge von Eingabewerten in x in eine ω-Folge von Ausgabewerten in y.* f soll bestimmte Spezifikation erfüllen.
Es gibt verschiedene Parameter in dieser Aufgabenstellung:

- Die Art, in der die Spezifikation gegeben ist.
- Der Wertebereich der Variablen x und y.
- Die Arbeitsweise von f. – Wir nennen f synchron, wenn zu jedem Eingabewert sofort ein Ausgabewert produziert wird. Ansonsten heißt f asynchron.

Ziel: Finde einen Algorithmus, der zu einer gegebenen Spezifikation ein Programm f liefert, das diese Spezifikation erfüllt (falls ein solches existiert). (Das Problem geht zurück auf Church (1963): "Church's problem".)

Im allgemeinen Fall ist diese Aufgabe nicht lösbar. Hier betrachten wir:

- LTL-Spezifikationen
- Variablen mit endlichem Wertebereich
- synchrone Programme

Formalisierung: Wir bezeichnen die Wertebereiche von x und y mit D_x und D_y. Ein Programm kann dann als Funktion $f : D_x^+ \to D_y$ aufgefaßt werden. Im i-ten Schritt liefert f eine Ausgabe, die von den ersten i Eingaben abhängen kann:

$(x_0, y_0)(x_1, y_1)(x_2, y_2)$ mit $y_i = f(x_0 \dots x_i)$.

Eine Berechnung von f ist ein ω-Wort $\alpha \in (D_x \times D_y)^\omega$ mit $\alpha_y(i) = f(\alpha_x(0) \dots \alpha_x(i))$. Dabei stehen α_x und α_y jeweils für die Projektion auf die entsprechende Komponente. Die Menge der Berechnungen von f ist

$L(f) = \{\alpha \in (D_x \times D_y)^\omega | \alpha \text{ ist Berechnung von } f\}$.

Die LTL-Formeln für die Spezifikationen werden aufgebaut aus atomaren Formeln $x = d$ für $d \in D_x$ oder $y = d$ für $d \in D_y$ und Booleschen und temporalen Operatoren.

[Folie: "LTL"]

Beispiele:

- $D_x = \{req, \dots\}$, $D_y = \{ack, \dots\}$: $\varphi_1 = G(x = req \to F(y = ack))$ – Jedes req wird mit einem ack beantwortet.
- $D_x = \{req_1, req_2, rel_1, rel_2, \dots\}$, $D_y = \{ack_1, ack_2\}$: $\varphi_2 = \neg F(y = ack_1 \wedge (X(x \neq rel_1))U(y = ack2))$ – Zwischen ack_1 und ack_2 muß immer ein rel_1 kommen.

Wir nennen φ implementierbar, wenn ein Programm f existiert, das φ erfüllt, also mit $L(f) \subseteq L(\varphi)$. f heißt Implementierung von φ.

Beispiel: $D_x = D_y = \{1, 2\}$:

$$\varphi = (x = y \wedge G(x = 1 \rightarrow F(y = 1)) \wedge G(y = 2 \rightarrow (y = 2)U(x = 1))$$

Zwei mögliche Implementierungen:

- f_1: Setze Ausgabe immer gleich Eingabe: $x = y$
- f_2: Setze $y = 2$, solange $x = 2$. Ist zum ersten Mal $x = 1$, setze $y = 1$ für immer.

f_1 und f_2 können durch endliche Automaten mit Ausgabe dargestellt werden. Die Transitionen werden mit den Eingaben beschriftet und die Zustände mit den Ausgabetransitionen.

[Grafik]

Das Syntheseproblem lautet nun:

- Gegeben: LTL-Formel φ (über Variablen x, y)
- Gesucht: Programm f, das φ implementiert (falls so ein Programm existiert).

Wir untersuchen zunächst die Probleme, die sich ergeben, wenn man versucht, die Implementierbarkeit und Erfüllbarkeit von φ in Beziehung zu setzen.

Versuch 1: Ist φ implementierbar, wenn φ erfüllbar ist ?
Da φ Anforderungen an die Eingabe stellen kann, gilt dieser Zusammenhang nicht. Betrachte z.B. $X(x = 1)$. Da das Programm keinen Einfluß auf die Eingabe hat, wird es immer eine Berechnung der Form $(x_0, y_0)(2, y_1) \ldots$ geben. Also ist diese Formel nicht implementierbar. Allerdings ist sie erfüllbar (jede Folge, deren zweite Eingabe 1 ist, ist ein Modell der Formel). Das Problem ist, daß es Eingabefolgen gibt, die keine "passenden" Ausgabefolgen erlauben.

Versuch 2: Ist φ implementierbar, wenn für alle $\alpha \in D_x^\omega$ ein $\beta \in D_y^\omega$ existiert, mit

$$\alpha \times \beta \vDash \varphi ? \quad [\alpha \times \beta = (\alpha(0), \beta(0))(\alpha(1), \beta(1)) \ldots]$$

Betrachte $\varphi = (y = 1) \leftrightarrow X(x = 1)$. φ ist nicht implementierbar, da zur Berechung $(1, 1)(1, y_1) \ldots$ auch eine Berechnung $(1, 1)(2, y_1') \ldots$ existiert, die φ nicht erfüllt. Allerdings gibt es zu jeder Eingabefolge eine passende Ausgabefolge, nämlich die, die die erste Ausgabe gleich der zweiten Eingabe setzt. Um diese Schwierigkeiten zu vermeiden, werden wir von Wörten auf Bäume ausweichen. Wenn man $k = |D_x|$ wählt, kann man o.B.d.A. davon ausgehen, daß $D_x = \{1, \ldots, k\} = [k]$. Erweitert man die Definition eines Programmes f auf $[k]^*$ (setze $f(\varepsilon) =$ beliebig), dann ist f ein Baum. Eine Formel φ ist implementierbar, wenn es einen Baum $t \in T^\omega_{D_y,k}$ gibt, sodaß für jeden Pfad $\pi = x_0 x_1 \ldots \in [k]^\omega$ gilt:

$$C_t(\pi) = (x_0, t(x_0)) \ (x_1, t(x_0 x_1)) \ (x_2, t(x_0 x_1 x_2)) \ldots \vDash \varphi$$

Die Bäume zu f_1, f_2 aus obigem Beispiel:

[Grafik]

Wir schreiben auch t implementiert φ und setzen $T(\varphi) = \{t \in T^\omega_{D_y,k} | t \text{ implementiert } \varphi\}$.

Ziel: Konstruiere PBA, der $T(\varphi)$ akzeptiert.

Vorgehen: $\varphi \rightarrow$ nichtdeterministischer Büchi-Wortautomat $\mathfrak{A}_1$ mit $L(\varphi) = L(\mathfrak{A}_1) \rightarrow$ deterministischer Paritätswortautomat (PWA) $\mathfrak{A}_2$ mit $L(\mathfrak{A}_1) = L(\mathfrak{A}_2) \rightarrow$ PBA $\mathfrak{A}_3$ mit $T(\mathfrak{A}_3) = T(\varphi)$.

1. Schritt:

Satz: Zu jeder LTL-Formel φ exsitiert ein nichtdeterministischer BWA, der $L(\varphi)$ erkennt.

Beweis:

a) Andere Vorlesung (Automaten auf unendlichen Wörtern).

b) LTL ist ein Spezialfall auf 1-verzweigten Bäumen: $\Gamma = \{1\}$

- $X\varphi \to \langle 1 \rangle \varphi$
- $F\varphi \to \langle 1^* \rangle \varphi$
- $\varphi_1 U \varphi_2 \to \langle (\varphi_1?; 1)^* \rangle \varphi_2$
- $G\varphi = \neg F \neg \varphi$

2. Schritt:

Satz: Zu jedem nichtdeterministischen BWA mit n Zuständen existiert ein äquivalenter deterministischer PWA mit $2^{O(n \log n)}$ Zuständen und $O(n)$ Prioritäten.

Beweis: BWA n Zustände $\to$ deterministischer Rabin-Automat $2^{O(n \log n)}, O(n)$ Paare $\to$ deterministischer PWA $2^{O(n \log n)}$ Zustände, $O(n)$ Prioritäten

q.e.d.

3. Schritt:

Lemma: Zu jedem deterministischen PWA $\mathfrak{A}$ über $([k] \times D_y)$ existiert ein PBA $\mathfrak{A}'$, dessen Größe linear ist in der Größe von $\mathfrak{A}$, sodaß

$$T(\mathfrak{A}') = \{t \in T^\omega_{D_y,k} | \forall \pi \in \Pi_k : C_t(\pi) \in L(\mathfrak{A})\}.$$

$\pi = i_1 i_2 i_3 \ldots$ mit $i_j \in [k]$ und $C_t(\pi) = (i_1, a_1), (i_2, a_2) \ldots$

Beweis: Idee: Eine Transitionsfolge $q_0 \overset{(i_1,a_1)}{\to} q_1 \overset{(i_2,a_2)}{\to} q_2 \ldots$ in $\mathfrak{A}$ entspricht in $\mathfrak{A}'$:

[Grafik]

Ist $\mathfrak{A} = (Q, [k] \times D_y, q_{in}, \delta, c)$, so definieren wir $\mathfrak{A}' = (Q', D_y, \{q'_{in}\}, \Delta', c')$ durch

- $Q' = \{q'_{in}\} \cup (Q \times [k])$ mit q'_{in} neuer Zustand
- $\begin{aligned} \Delta' &= \{(q'_{in}, a, (q_{in}, 1), \ldots, (q_{in}, k)) | a \in D_y\} \\ &= \{((q,i), a, (\delta(q,(i,a)), 1), \ldots, (\delta(q,(i,a)), k) | q \in Q, iin[k], a \in D_y\} \end{aligned}$
- $c'(q'_{in} = 0$ und $c'(q,i) = c(q)$.

Dann gilt $T(\mathfrak{A}') = \{t \in T^\omega_{D_y,k} | \forall \pi \in \Pi_k : C_t(\pi) \in L(\mathfrak{A})\}$.

q.e.d.

Satz: Zu jeder LTL-Formel $\varphi(x, y)$ kann ein PBA $\mathfrak{A}$ konstruiert werden mit $T(\mathfrak{A}) = T(\varphi)$ und $2^{2^{O(|\varphi|)}}$ Zuständen und $2^{O(|\varphi|)}$ Prioritäten.

Beweis: $\varphi \to$ nichtdeterministischer BWA mit $2^{c_1 \cdot |\varphi|}$ Zuständen $\to$ deterministischer PWA mit

$$\begin{aligned} &2^{c_2 \cdot (2^{c_1 \cdot |\varphi|} \cdot \log 2^{c_1 \cdot |\varphi|})} \\ &\leq 2^{c_2 \cdot (2^{2c_1 \cdot |\varphi|})} \\ &\in 2^{2^{O(|\varphi|)}} \end{aligned}$$

Zuständen.

q.e.d.

Satz: Zu jeder LTL-Formel $\varphi(x, y)$ kann in doppelt exponentieller Zeit entschieden werden, ob sie implementierbar ist. Desweiteren kann eine Implementierung von φ konstruiert werden, die durch einen endlichen Automaten mit Ausgabe darstellbar ist.

Beweis: Der PBA $\mathfrak{A}$ mit $T(\varphi) = T(\mathfrak{A})$ kann in Zeit $2^{2^{O(|\varphi|)}}$ konstruiert werden. Der Leerheitstest für $\mathfrak{A}$ ist in Zeit $\left(2^{2^{O(|\varphi|)}}\right)^{2^{O(|\varphi|)}} = 2^{2^{O(|\varphi|)}}$ durchführbar. Ist $T(\mathfrak{A}) = \varnothing$, so liefert eine positionale Strategie in $G_{\mathfrak{A}}$ einen regulären Baum in $T(\mathfrak{A})$. Dieser Baum aufgefaßt als Programm ist eine Implementierung von φ.

q.e.d.

Beispiel: $\varphi = (x = y) \wedge G(x = 1 \rightarrow F(y = 1)) \wedge G(y = 2 \rightarrow (y = 2)U(x = 1))$; deterministischer Büchi-Wortautomat für φ:

[Grafik]

Der entsprechende Baumautomat hat 11 Zustände und 22 Transitionen. Die Übersetzung in den Baumautomaten und das Lösen des Leerheitsspiels $G_{\mathfrak{A}}$ entspricht in diesem Fall der Auswahl einer Transition für jede mögliche Eingabe von jedem Zustand aus, sodaß in dem resultierenden Automaten alle Läufe akzeptierend sind.
Ändere φ_2 wie folgt: $(y = 2)U(x = 1 \vee G(x = 2))$

ANMERKUNG: Das Programm f_1 aus der letzten Vorlesung ist keine Implementierung von φ.

Vorteil der Synthese: Zu einer Spezifikation wird sofort ein Programm geliefert, das die Spezifikation erfüllt, während beim Model-Checking zuerst ein System entworfen, dann verifiziert und ggf. korrigiert wird.

Nachteil der Synthese:

- Wesentlich höhere Komplexität
- Man muß alle Anforderungen an das System in die Spezifikation schreiben, das System also komplett in einer Logik formalisieren. Beim Model-Checking können Teilaspekte (z.B. nur sicherheitsrelevante Aspekte) verifiziert werden.

Kompromiß: "Controller-Synthese" Dabei wird das System nicht komplett entworfen, sondern enthält noch nichtdeterministische Verzweigungen. Ziel ist es, dieses nichtdeterministische System so einzuschränken, daß es der Spezifikation genügt.
Idee: Nichtdeterministisches System S und deterministischer Controller $C \rightarrow$ deterministisches System $S \times C$, das die Spezifikation erfüllt

Ausblick: Verteilte Synthese Statt eines Programmes sollen mehrere Programme konstruiert werden, die miteinander kommunizieren können (über Variablen). Die Spezifikation spricht über Ein- und Ausgabevariablen. Das Syntheseproblem für eine solche Architektur ist:

- Gegeben: Wertebereiche für die Variablen und eine Spezifikation φ
- Gesucht: Programme für die Architektur, sodaß das Gesamtsystem die Spezifikation erfüllt

Dieses Problem ist für die meisten Architekturen unentscheidbar. Entscheidbar nur für "Pipelines":

$x \rightarrow \bigcirc \rightarrow \bigcirc \rightarrow \ldots \rightarrow \bigcirc \rightarrow y$